Rapport : Projet Graphique 3D

Auteurs

- Loïc SCHAMBER
- Simon POPELIER
- Yohan GOUZERH

Objectif

Dans ce tp nous avons voulu recréer une scène très connue du crétacé : l'arrivée de terribles météorites dans le monde idyllique des dinosaures. Il se trouve qu'un dinosaure charpentier aurait également créé une porte.

Caméra

Pour contrôler la caméra et ainsi se déplacer dans le monde, nous avons récupéré la trackball et ajouté des fonctions qui permettent de la translater sur tous les axes et de la tourner. La translation fonctionne bien, cependant la rotation n'est possible que autour de l'origine sur l'axe y. Réussir à effectuer cette rotation nous permettrait de suivre un dinosaure par exemple.

Accessoires

Dinosaures

Nous avons utilisé le modèle animé du tp7 ainsi qu'un modèle de T-rex fixe téléchargé. Sur le T-rex nous avons appliqué une texture. Sur le dinosaure animé, dès que nous appuyons sur une touche qui peut déplacer, nous vérifions si le cycle d'animation précédent est terminé. Si oui, il reset l'offset dédié au noeud, sinon il ne fait rien, pour avoir une animation continue et qui s'arrête quand le dinosaure s'arrête. Le dinosaure avance avec les touches : "y", "g", "h" et "j".

Rochers

Nous avons créé une classe Rocher qui permet de créer de façon procédurale un objet fermé de qualité douteuse.

L'algorithme utilisé comporte des parties assez simples. - Dans un premier temps, on part d'un tétraèdre. - Une itération consiste ensuite à complexifier ses faces en rajoutant un point au dessus d'un de ses triangles et de former alors 3 nouveaux triangles. Cette étape peut être répétée un certain nombre de fois mais s'avère rapidement néfaste car l'objet se recroqueville alors sur lui (à cause de la composante aléatoire). - Une autre étape consiste à décomposer chaque triangles en 16 autres triangles pour pouvoir légèrement perturber les points centraux. Cela permet de rendre les faces moins lisses.

Une fois encore, pour des raisons de facilité, l'objet est une liste de points utilisant la première

méthode montrée dans le TP2 soit sans index. Ce choix permet d'éviter la difficulté du stockage des indexes mais par la suite est un frein pour la perturbation des points. En effet, perturber un point revient à effectuer la même perturbation pour toutes les faces dont il fait partie...

Pour ce qui est des normales et donc de l'éclairage, nous avons appliqué le même principe que pour le terrain.

Arbres

Dans le processus de création de rochers, nous nous sommes aperçus que la classe créée permettrait également à quelques modifications près de simuler une masse de feuilles.

Les paramètres que nous avons rajouté au rochers pour en faire une masse de feuilles plus réaliste sont nombreux mais consiste beaucoup à mettre des bornes aux longueurs des arrêtes générées au fur et à mesure ainsi qu'à choisir le nombre d'itération de la première étape de l'algorithme décrit précédemment.

Nous en avons profité pour créer également des arbres procéduraux. Cela consiste à ajouter un tronc sous la forme d'un simple cylindre. Il faut alors l'orienter vers la masse de feuilles. Nous réalisons cela de manière approximative en donnant le centre du tétraèdre initial et en plaçant le cylindre relativement à ce point.

Le tronc est ensuite éclairé de la même manière que la masse de feuilles.

On peut déplorer que tous les arbres et rochers du terrains, bien que générés aléatoirement, soient les mêmes par manque de temps. Une correction minime aurait été suffisante pour diversifier l'environnement.

Porte

Nous avons ajouté une porte contrôlable au clavier. Elle s'ouvre grâce aux touches "l" et "p" et pivote par rotation contrôle node. Elle est symétrique et comporte donc 2 cylindres pour l'axe de rotation et deux pavés aplatis pour la porte en elle-même.

Météorites

Pour implémenter un objet avec des keyframes, nous avons pensé faire des météorites qui tomberaient simplement depuis un endroit aléatoire dans le ciel, jusqu'au sol. Il suffit de donner 2 keyframes pour le point d'apparition et le point de chute (point de chute qui peut être sous la map). De plus, nous avons modifié la classe KeyframeControlNode afin de pouvoir spécifier une durée après laquelle l'animation recommence, afin de pouvoir profiter de la pluie de météorites en continu.

Terrain

Génération du terrain

Nous avons choisi de générer notre terrain à partir d'une heightmap.

À partir de l'image, nous obtenons une matrice d'une image en niveau de gris. La valeur du gris en un point donne ensuite la hauteur du terrain. La position dans la matrice nous donne les coordonnées restantes.

Un carré de base défini par 4 points de la matrice est divisé en 2 triangles affichables.

Traitements :

Nous avons effectué plusieurs traitements sur le terrain pour le rendre plus manipulable par la suite.

Tout d'abord, il est normalisé. L'amplitude de sa hauteur est ramenée entre 0 et 1. Cela nous permet par la suite de définir sa hauteur finale en multipliant par un même facteur la hauteur de tous les points.

La longueur des arcs est également changeable, permettant de rétrécir ou augmenter la largeur et longueur du terrain.

Couleurs :

La couleur d'un point du terrain est donnée par sa hauteur. Nous avons choisi un simple dégradé entre 2 couleurs voulues (la couleur du point le plus bas et celle du point le plus haut).

Pour ce qui concerne les ombres du terrain. Pour des raisons de facilité, nous avons choisi de donner les normales des faces plutôt que celles des vertex. Ainsi le rendu de l'illumination est low-poly. Un point possède une composante ambiante et une composante illuminée.

Mise en place de la scène

Generation des objets

Pour positionner les objets du terrain, notre classe Map s'occupe de générer tous les nodes de la map. Nous avons modifié les nodes pour contenir trois matrices T/R/S au lieu d'une matrice transform, permettant d'être plus flexible, de récupérer les opérations effectuées, de supprimer des translations, rotations,... Cela nous a permis de réaliser dans la classe node, une api qui nous permet d'effectuer des traitements sur l'objet de type translate/rotate/scale (api relatif à la position ou global dans la scène) La plupart des éléments, comme les arbres et les rochers sont générés aléatoirement sur la map grâce à cette api.

Mettre sur le terrain à la bonne hauteur

Pour placer les objets sur le sol à la bonne hauteur nous récupérons la hauteur du point le plus proche. Nous utilisons une hashmap avec comme clef la position en 2d sur x et z du point, pour réaliser cette recherche en O(1) au lieu de reparcourir à chaque fois toutes les vertex du terrain. Ensuite, nous mettons à jour la position de l'objet avec cette hauteur. Pour être plus fidèle et ne pas dépendre du nombre de vertex, nous pourrions utiliser un algorithme d'intersection du plan

(triangle formé par les 3 vertex les plus proches) et de la droite y pour récupérer cette hauteur.

Skybox

Notre skybox est un grand cube sur lequel nous avons appliqué une texture de ciel sur les faces intérieures.

Textures

Pour certains de nos accessoires, nous voulions réaliser une texture de Perlin. Cependant, nous n'avons pas réussi à aboutir à une texture satisfaisante. En effet, le bruit de Perlin s'effectue en 3 principales étapes. 1: Generer du bruit, 2:interpoler le bruit pour le rendre plus doux, 3: creer le bruit de perlin a proprement parler en utilisant plusieurs bruits obtenus a l'étape deux pour obtenir cet effet "nuageux" distinctif du bruit de Perlin. Nous avons rencontré des problèmes lors de l'interpolation en 2D. Nous voulions effectuer une interpolation 2D avec cosinus, sans succès. Nous avons cependant réussi à obtenir une partie 1 correcte pour du simple bruit blanc et pour la partie 3 qui consiste à additionner la texture à plusieurs fréquences.